

The SPC Technique and Low Complexity Turbo Codes

Li Ping, Dept. of EE, City University of Hong Kong, Hong Kong, email: eeliping@cityu.edu.hk

Abstract

This paper proposes a family of modified turbo codes. The decoding cost, in terms of both operation number and storage usage, of the proposed scheme is significantly lower (5~10 times) than the standard turbo codes with similar performance. The scheme can also be used to alleviate the error floor problem and is very flexible for median to high rate code designs.

I. Introduction

This paper is concerned with a family of modified turbo codes [1], referred to as the SPC-turbo codes [2]. An SPC (Single Parity Check) technique is introduced to replace puncturing commonly used in the turbo codes for rate adjustment. This greatly reduces the lengths of the convolutional codes involved. Very simple constituent codes, say 4-state ones, are used and performance is maintained by a multi-dimensional concatenation technique [3,4]. The decoding complexity, in terms of both operation number and memory usage, of an SPC-turbo code is only a fraction of that of a standard turbo code with comparable performance. For various median to high rates, the SPC-turbo codes can achieve BER=10⁻⁵ at about 0.5dB from the theoretical limits. The proposed scheme can also be used to improve the noise floor problem.

II. Encoding principles

The proposed code consists of two layers of concatenations, which are discussed separately.

A. First layer: the SPC-convolutional code

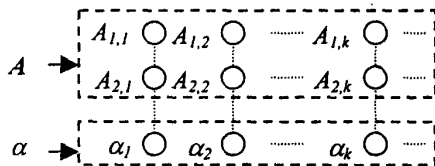


Fig.1. A is a binary array and α_k is the parity check of the k -th column of A .

Let $A = \{A_{j,k}\}$ be a binary array and A_k be the k -th column of A , Fig.1. Let α_k be the parity check of A_k ,

i.e., $\{A_k, \alpha_k\}$ form a SPC code. We call $\alpha = \{\alpha_k\}$ the column-parity-check vector (CPCV) of A .

The so-called SPC-convolutional code, denoted by C , is constructed as shown in Fig.2. The information bits of C are arranged into two arrays E and F . A codeword in C is of the form $\{E, F, q\}$. The redundant vector q is generated as follows.

- Let p be the CPCV of E .
- p is used to drive a specified systematic convolutional code \hat{C} , producing a redundant vector p' .
- q is the CPCV of $\begin{bmatrix} F \\ p' \end{bmatrix}$.

We can treat \hat{C} as a rate 1/2 convolutional code $p \rightarrow (p, p')$. Notice that p and p' do not appear in the final codeword but their information is carried in E and $\begin{bmatrix} q \\ F \end{bmatrix}$. A more concise definition follows.

Definition 1: SPC-convolutional code

- Arrange a codeword as $A = \begin{bmatrix} E & F \\ p & q \end{bmatrix}$. (A can have variable column sizes.)
- The CPCV of A , $\alpha = \{\alpha_k\} = \{p, p'\}$, is in a specified rate 1/2 systematic convolutional code \hat{C} .

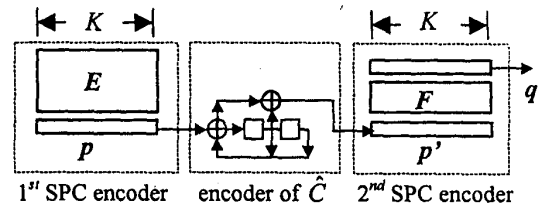


Fig.2. An SPC-convolutional encoder. The codeword is $A = \{E, F, q\}$ with q containing the redundant bits. The CVCA of A is $\alpha = (p, p') \in \hat{C}$.

Let E and F have K columns each. Let J_E and J_F be their row numbers respectively and denote $J = J_E + J_F$. The information length of C is $J \times K$. The rate of C is

$$J/(J+1) \tag{1}$$

B. Second layer: the overall SPC-turbo code

Fig.3 illustrates an M -dimensional SPC-turbo encoder. The information sequence D is interleaved and encoded M times using the encoder in Fig.2. Let $q^{(m)}$ be the redundant vector of the m -th encoder. The overall codeword is formed by $\{D, q^{(1)}, \dots, q^{(M)}\}$. The pair $\{D, q^{(m)}\}$ is referred to as the m -th dimension. Based on (1), the overall rate of this code is

$$R = J/(J+M) \quad (2)$$

that is adjustable. The circular technique of [5] is employed to handle the terminations.

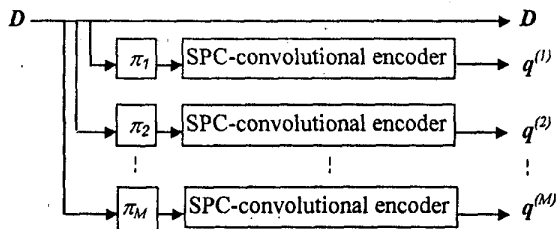


Fig.3. The encoder of an SPC-turbo code where $\{\pi_m\}$ are interleavers. The SPC-convolutional encoder has the structure as given in Fig.2.

C. Trellis length

The total trellis length of the SPC-turbo code (including all the dimensions) is

$$L = M \times K \quad (3)$$

As shown later, L approximately determines the decoding costs. Combining (2) and (3), we have

$$L = (R^{-1} - 1) \times J \times K \quad (4)$$

Two observations can be made.

- L reduces for increasing R .
- Fixing R , L is independent of M .

These lead to the low decoding cost and low error floor properties of the code, as detailed later.

III. Decoding method

This section addresses the decoding issue for the SPC-turbo code. We first describe an efficient three-step APP (*a posteriori* probability) decoding principle for the constituent SPC-convolutional codes. We then outline an iterative decoding strategy for the overall SPC-turbo codes.

A. APP decoding of the SPC-convolutional code

Let $c = \{c_i\}$ be the codeword in the BPSK format $\{0 \leftrightarrow 1, 1 \leftrightarrow -1\}$ and $x(c) = \{x_i\}$, or simply x , be the noisy observation of c . Define the *a priori* LR (Likelihood Ratio) of a bit c_i conditioned on x_i as

$$\tilde{L}_i \equiv \tilde{L}(c_i) \equiv \frac{\Pr\{c_i = +1 | x_i\}}{\Pr\{c_i = -1 | x_i\}} \quad (5)$$

This is equivalent to the definition of LLR (Logarithm of Likelihood Ratio) [1,6] except that we have dropped logarithm for convenience. The APP decoding is to find the *a posteriori* LR for c_i as,

$$L(c_i | x) \equiv \frac{\Pr\{c_i = +1 | x\}}{\Pr\{c_i = -1 | x\}} \quad (6)$$

Denote x excluding y by $x \setminus y$, with $y \supset x$. Assuming that distortions are independent, (6) can be expressed as,

$$L_i(x_i | x) = \tilde{L}_i E_i \quad (7)$$

where

$$E_i \equiv E(c_i | x \setminus x_i) \equiv \frac{\Pr\{c_i = +1 | x \setminus x_i\}}{\Pr\{c_i = -1 | x \setminus x_i\}} \quad (8)$$

is called the extrinsic LR for c_i provided by $x \setminus x_i$ [1].

Eqn.(6) can be evaluated for a code defined in Definition 1 using a three-step algorithm listed below. It is a generalisation of the two-way scheduling discussed in [7]. Initially, we set $\tilde{L}(\alpha_k) = 1$ (since there is no observation of $\{\alpha_k\}$). For simplicity, “decode” below is in the APP sense. Notice that the algorithm performs an exact APP decoding.

Algorithm 1:

Step 1: Find $E(\alpha_k | A_k)$ for every k .

Step 2: Use the results of Step 1 as the *a priori* LR for $\{\alpha_k\}$. Decode \hat{C} and find the extrinsic LR for $\{\alpha_k\}$.

Step 3: Decode $A_k \cup \alpha_k$ using the results of Step 2 as the *a priori* LR for α_k .

In the above, Step 2 can be implemented by the standard MAP algorithm [1]. We now give simple rules for implementing the other two steps.

Implementation of Step 1

Let a be a binary sequence over $\{-1, +1\}$. Define the *plr* (parity likelihood ratio) function

$$plr(a) \equiv \frac{\Pr\{a \text{ even} | x(a)\}}{\Pr\{a \text{ odd} | x(a)\}} \quad (9)$$

where “ a even” (resp. “ a odd”) is the event that a contains an even (resp. odd) number of -1 . The *plr* function can be evaluated as follows. If a contains only one bit a_1 , then $plr(a) = \tilde{L}(a_1)$. Otherwise a can

be partitioned into two non-overlapping subsets b and c . It can be verified that

$$plr(a) = \frac{plr(b) \times plr(c) + 1}{plr(b) + plr(c)} = f(plr(b), plr(c)) \quad (10a)$$

$$\text{with } f(x,y) = (xy+1)/(x+y) \quad (10b)$$

Back to Algorithm 1. Since α_k is the parity check of A_k , we have

$$E(\alpha_k | x(A_k)) = \frac{Pr(\alpha_k = +1 | x(A_k))}{Pr(\alpha_k = -1 | x(A_k))} = plr(A_k) \quad (11)$$

Based on (10), $plr(A_k)$ can be evaluated recursively from smaller set of A_k . Start from $f_1 = \tilde{L}(A_{1,k})$. Let

$$f_j = f(f_{j-1}, \tilde{L}(A_{j,k})) \text{ for } j=2, 3, \dots, r \quad (12)$$

where r is the size of A_k . Then $plr(A_k) = f_r$.

Implementation of Step 3

Use the result of Step 2 as the *a priori* LR for α_k . Step 3 involves the decoding of $A_k \cup \alpha_k$. Let

$$\begin{aligned} b &= \{A_{1,k}, A_{2,k}, \dots, A_{j-1,k}\} \\ c &= \{A_{j+1,k}, \dots, A_{r,k}, \alpha_k\} \\ a &= b \cup c = \{A_{1,k}, \dots, A_{j-1,k}, A_{j+1,k}, \dots, A_{r,k}, \alpha_k\} \end{aligned} \quad (13)$$

Combining (7), (10) and (11), for any $A_{j,k}$ in A_k

$$L(A_{j,k}) = \tilde{L}(A_{j,k}) plr(a) = \tilde{L}(A_{j,k}) f(plr(b), plr(c)) \quad (14)$$

Here $plr(b) = f_{j-1}$ has already been generated in Step 1 and $plr(c)$ can be found using a simple recursion. Start from $e_r = E(\alpha_k | x(A_k))$ and compute

$$e_j = f(e_{j+1}, \tilde{L}(A_{j,k})) \text{ for } j=r, r-1, \dots, 2 \quad (15)$$

Then $plr(c) = e_{j+1}$.

Complexity analysis

Step 1 costs $r-1$ f -functions per column. If A_k contains information bits only (such as a column in E), Step 3 costs r multiplications and $2(r-1)$ f -functions per column. If A_k contains a parity check bit (such as a column in $\begin{bmatrix} q \\ F \end{bmatrix}$) which does not require

output, Step 3 costs $r-1$ multiplications and $2r-3$ f -functions per column.

B. Iterative decoding of the overall code

Fig.4 shows a global iterative decoder [2] for the overall SPC-turbo code. It consists of a loop of M local APP decoders, each corresponding to one dimension. The variables are explained below.

- \tilde{L} is the *a priori* LR vector of information bits.
- $\tilde{L}(q^{(m)})$ is the *a priori* LR vector for $q^{(m)}$.

- $L^{(m)}$ is the *a posteriori* LR vector of information bits generated by the m -th APP decoder.
- $E^{(m)}$ is the extrinsic LR vector of information bits generated by the m -th APP decoder. It is related to \tilde{L} and $L^{(m)}$ by (7), i.e., $L_i^{(m)} = \tilde{L}_i E_i^{(m)}$.

The decoder in Fig.4 is a multi-dimensional generalisation of the standard turbo decoder [1,4]. At start, \tilde{L} , calculated from the individual observations, is fed to DEC-1. The M decoders are operated successively along the loop. The *a posteriori* LR values generated by a local decoder are used as the *a priori* values for the next one.

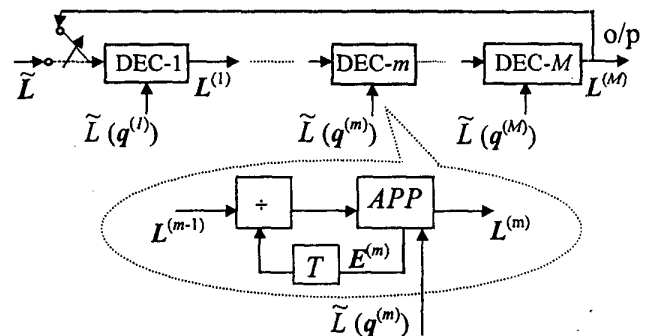


Fig.4 An M -dimensional decoder. The block labelled by T represents delay of one iteration. The switch is at the input position initially and at the feedback position afterwards. Initially, $\{E_i^{(m)}\}$ are set to 1.

It is important to note the function of the block labelled by the division sign \div . It stands for the bit-by-bit division between the feedback $L^{(m)}$ and the stored extrinsic LR $E^{(m)} = \{E_i^{(m)}\}$ (equivalent to subtraction in the log domain [1]). It prevents, to a large extent, the outputs of a local decoder from circulating back to itself again. A detail discussion can be found in [4].

IV. Numerical results

Let M be the dimension number and N the state number. We observed by simulation that a good trade-off between decoding complexity and performance is achieved by setting $M=3$ and $N=4$. As shown in Fig.5 for rate $R=1/2$, the performances of the SPC-turbo codes with $(J_E, J_F) = (2, 1)$ are almost the same as the 2-dimensional, 16-state standard punctured turbo codes.

The interleavers used in the simulations are random except that every bit should appear in E and F for exactly $J_E \times M/J$ and $J_F \times M/J$ times respectively. For example, with $M=3$ and $(J_E, J_F)=(2,1)$, every bit should appear twice in E and once in F . This can be easily realised by properly grouping the information bits for E and F in every dimension.

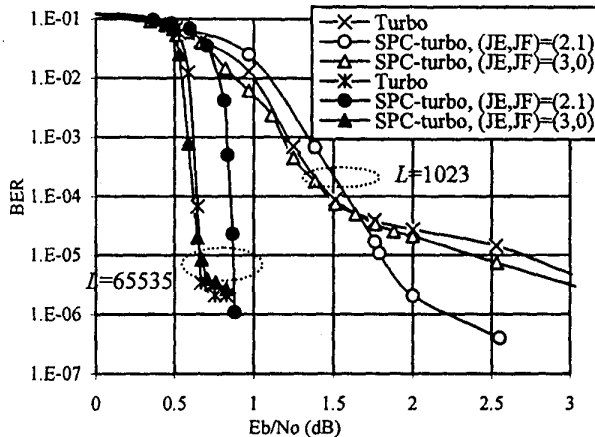


Fig.5. Performance comparison between the SPC-turbo and standard turbo codes for rate $R=1/2$ and information length = 1024 and 65535. For the SPC-turbo code, $M=3$, $N=4$, and \hat{C} is given by $(1+x)/(1+x+x^2)$. For the standard turbo code, $M=2$, $N=16$ and transfer function is $(1+x^4)/(1+x+x^2+x^3+x^4)$.

The performances of $(J_E, J_F)=(3,0)$ SPC-turbo codes is also included in Fig.5, which exhibit relatively lower error floor. Generally speaking, increasing the size of E leads to lower error floor. This is because E is followed by a recursive convolutional encoder, which introduces an interleaving gain. However, increasing the size of F can improve performance in the so-called waterfall range (where the BER having a steep drop). We are currently looking into this issue.

During an iterative decoding process, there can be information loss at the interfaces of the local decoders. This may affect the convergence speed of the multi-dimensional decoder. However, as seen from Fig.6, the difference between 2- and 3-dimensional decoders is less than one iteration, which is negligible when iteration number is more than 10.

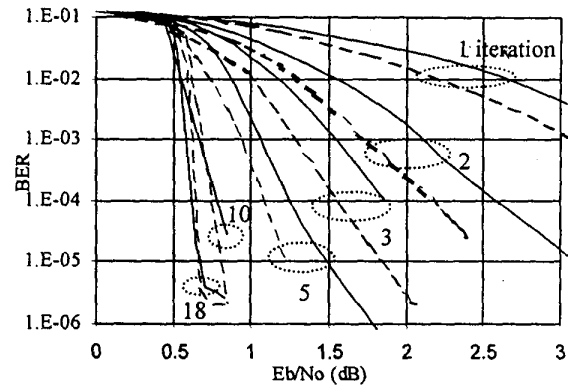


Fig.6. Convergence comparison for the SPC-turbo and standard turbo codes used in Fig.5 with information length = 65535. The solid lines are for the SPC-turbo code and the dashed lines for the standard turbo code. The labels are iteration numbers.

The performances of the SPC-turbo codes with various rates are shown in Fig.7. In all the cases, the performances are only about 0.5 dB away from the theoretical limits, which are similar to those of the standard turbo codes with puncturing [8].

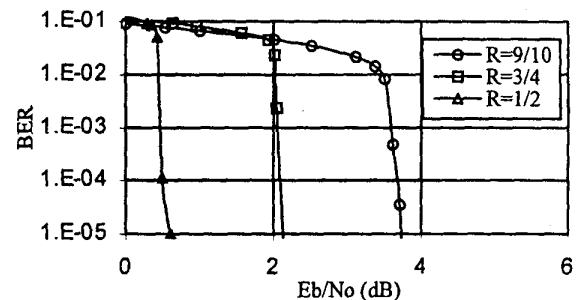


Fig.7. Performances of the SPC-turbo codes with various rates. The parameters are: $M=3$, $N=4$. $J \times K \approx 65535$ (or nearest integers divisible by $J=J_E+J_F$). The values for (J_E, J_F) are: $(2,1)$ for $R=1/2$, $(6,3)$ for $R=3/4$ and $(9,18)$ for $R=9/10$. Theoretical limits: $(R, E_b/N_0) = (1/2, 0.2\text{dB})$, $(3/4, 1.63\text{dB})$ and $(9/10, 3.2\text{dB})$.

Some other observations are:

- With state number $N=4$, increasing the dimension number M to more than 3 can reduce noise floor but cannot improve the waterfall range.

- With dimension number $M=3$, increasing the state number N to more than 4 can reduce noise floor but cannot improve the waterfall range.
- With state number $N=2$, the best choice of dimension number is $M=4$. However, the performance is somehow worst that those shown in Fig.5 and 7.

Complexity comparisons

The comparison below is made between $N=4$, $M=3$ SPC-turbo codes and $N=16$, $M=2$ standard turbo codes. As shown earlier, these two families of codes have nearly the same performances.

We estimate the decoding complexity of the MAP algorithm following [9]. Including normalisation, the cost is $8N$ multiplications and $6N$ additions per trellis section. Therefore, setting $N=4$, Step 2 of Algorithm 1 requires about 32 multiplications and 24 additions per trellis section. For $R=1/2$, L equals information length (see (4)) so cost per trellis section equals cost per information bit.

With $R=1/2$, $M=3$ so $J=3$. The combined costs of Steps 1 and 3 are 2 f -functions and 2 multiplications per information bit per dimension (division counted as multiplication). Including Step 2, the normalised cost of Algorithm 1 is about $32+3\times 6=50$ multiplications and $24+3\times 4=36$ additions per information bit per iteration.

As comparison, a 2-dimensional, 16-state turbo code requires about $2\times 8\times 16=256$ multiplications and $2\times 6\times 16=192$ additions per information bit per iteration regardless of rate. It is seen that the decoding cost of a rate $1/2$ SPC-turbo code is about 5 times lower than a comparable standard turbo code.

Notice that from (4), when rate R increases, the trellis length reduces and so does the cost involved in Step 2 of Algorithm 1. For $R\rightarrow 1$, the decoding cost of a SPC-turbo code is about 10 times lower than that of a punctured turbo code.

The costs of Steps 1 and 3 of Algorithm 1 increase with M , but the impact is small compared with Step 2. Since L is independent of M for fixed R (see (4)), the total decoding cost of a SPC-turbo code is only marginally affected by M for median rate codes. This feature is important since increasing dimension number can lead to lower error floor. As

comparison, the decoding complexity of a standard turbo code increases linearly with M .

The storage requirement can be a serious concern for long turbo codes, as the MAP algorithm needs to store $N\times K$ real values during the forward recursion, where K is the trellis length of a local decoder. For the standard turbo code, $N=16$ and K is the same as the information length. For the SPC turbo code, $N=4$ and K is only a fraction of the information length $J\times K$. The advantage of SPC-turbo code is clearly seen.

V. Conclusions

With comparable performances, the decoding complexities of the proposed SPC-turbo codes are significantly lower than the standard turbo codes. The proposed scheme also provides a low cost means to treat the error floor problem by adopting the multidimensional concatenation technique.

Acknowledgements

The author is grateful to J. Ho, K. Wu and W.K. Leung for their valuable help.

References

- [1] C. Berrou and A. Glavieux, "Near optimum error-correcting coding and decoding: Turbo-codes", *IEEE Trans. Commun.*, COM-44, pp.1261-1271, 1996.
- [2] Li Ping, "Modified turbo codes with low decoding complexity," *Electronics Letters*, vol. 34, no. 23, pp. 2228-2229, Nov. 1998.
- [3] D. Divsalar and F. Pollara, "Multiple turbo codes", *Proc IEEE Milcom-95*, pp.279-285, 1995.
- [4] Li Ping, S. Chan and K.L. Yeung, "Iterative decoding of multidimensional concatenated single parity check codes", *Proc. ICC-98*, pp. 136-140, 1998.
- [5] C. Berrou, "Some clinical aspect of turbo codes", *Proc. Int. Symposium on Turbo Codes*, France, pp.26-31, 1997.
- [6] J. Hagenauer, E. Offer and L. Papke, "Iterative decoding of binary block and convolutional codes", *IEEE Trans. Inf. Theory*, IT-42, pp.429-445, 1996.
- [7] F.R. Kschischang and B.J. Frey, "Iterative decoding of compound codes by probability propagation in graphic models", *IEEE, JSAC*, vol-16, no.2, pp.219-230, Feb. 1998.
- [8] O.F. Acikel and W.E. Ryan, "High rate turbo codes for BPSK/QPSK channels", *Proc. ICC-98*, pp. 422-427, 1998.
- [9] P. Robertson, E. Villebrun and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", *Proc. IEEE ICC-95*, pp. 1009-1013, June 1995.